

# Introduction to Machine Learning

## Regularization Introduction



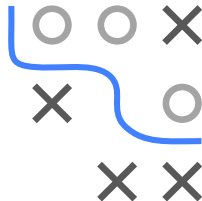
### Learning goals

- Overfitting
- Motivation of regularization
- First overview of techniques
- Pattern of regularized ERM formula



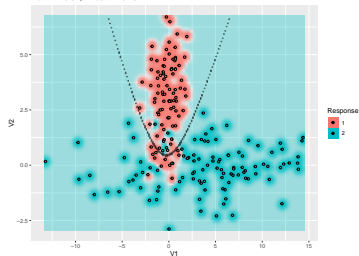
# RECAP: OVERFITTING

- Occurs when model reflects noise or artifacts in training data
- Model often then does not generalize well (small train error, high test error) – or at least works better on train than on test data



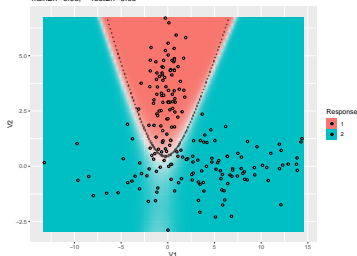
## Overfitted model

TrainErr=0.01; TestErr=0.13



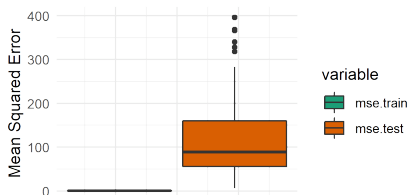
## Appropriate model

TrainErr=0.08; TestErr=0.06



# EXAMPLE 1: OVERFITTING

- Data set: daily maximum **ozone level** in LA;  $n = 50$
- 12 features: time (weekday, month); weather (temperature at stations, humidity, wind speed); pressure gradient
- Orig. data was subsetting, so it feels “high-dim.” now (low  $n$  in relation to  $p$ )
- LM with all features (L2 loss)
- MSE evaluation under  $10 \times 10$  REP-CV

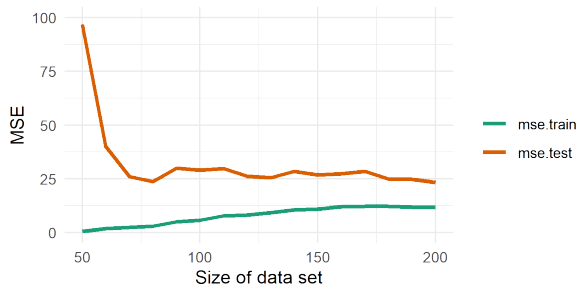


Model fits train data well, but generalizes poorly.



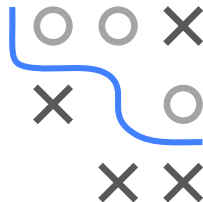
# AVOIDING OVERFITTING – COLLECT MORE DATA

We explore our results for increased dataset size.



Fit slightly worsens, but test error decreases.

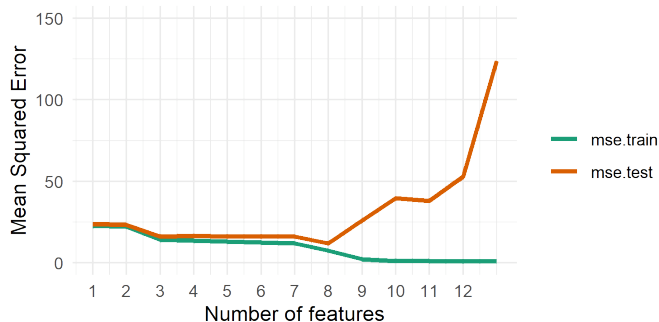
But: Often not feasible in practice.



# AVOIDING OVERFITTING – REDUCE COMPLEXITY

We try the simplest model: a constant. So for  $L2$  loss the mean of  $y^{(i)}$ .

We then increase complexity by adding one feature at a time.



NB: We added features in a specific (clever) order, so we cheated a bit.

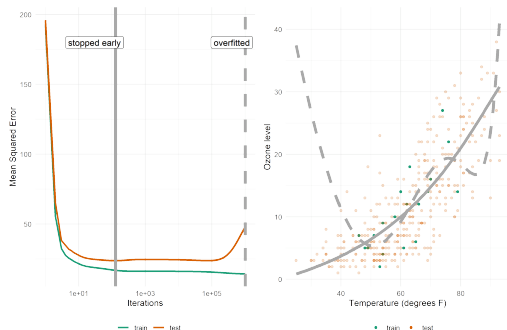


# AVOIDING OVERFITTING – OPTIMIZE LESS

Now: polynomial regression with temperature as single feature

$$f(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{k=0}^d \theta_k \cdot (x_T)^k$$

We set  $d = 15$  to overfit to small data. To investigate early stopping, we don't analytically solve the OLS problem, but run GD stepwise.



We see: Early stopping GD can improve results.

NB: GD for poly-regr usually needs many iters before it starts to overfit, so we used a very small training set.



# REGULARIZED EMPIRICAL RISK MINIMIZATION

We have contradictory goals:

- **maximizing fit** (minimizing the train loss)
- **minimizing complexity** of the model



We saw how we can include features in a binary fashion.  
But we would rather control complexity **on a continuum**.

# REGULARIZED EMPIRICAL RISK MINIMIZATION

Common pattern:

$$\mathcal{R}_{\text{reg}}(f) = \mathcal{R}_{\text{emp}}(f) + \lambda \cdot J(f) = \sum_{i=1}^n L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right) + \lambda \cdot J(f)$$

- $J(f)$ : **complexity penalty, roughness penalty** or **regularizer**
- $\lambda \geq 0$ : **complexity control** parameter
- The higher  $\lambda$ , the more we penalize complexity
- $\lambda = 0$ : We just do simple ERM;  $\lambda \rightarrow \infty$ : we don't care about loss, models become as “simple” as possible
- $\lambda$  is hard to set manually and is usually selected via CV
- As for  $\mathcal{R}_{\text{emp}}$ ,  $\mathcal{R}_{\text{reg}}$  and  $J$  are often defined in terms of  $\theta$ :

$$\mathcal{R}_{\text{reg}}(\theta) = \mathcal{R}_{\text{emp}}(\theta) + \lambda \cdot J(\theta)$$

